Process behavior meta-modeling for the derivation of enactment engines

Sana Damak Mallouli, Saïd Assar, Carine Souveyet

Process behavior meta-modeling for the derivation of enactment engines

Poster paper

Sana Damak Mallouli
Université Paris 1 Panthéon Sorbonne
Paris – France
sana.mallouli@malix-univ-paris1.fr

Saïd Assar
Institut Mines-Telecom,
Telecom Ecole de Management – France
said.assar@telecom-em.eu

Carine Souveyet
Université Paris 1 Panthéon Sorbonne
Paris – France
carine.souveyet@univ-paris1.fr

***Abstract*— To specify process modeling language semantics and build corresponding enactment engine is a challenging problem. We propose a model driven approach which combines structural and event-based behavioral meta-modeling techniques. These specifications are transformed into a software architecture for a process enactment engine that exploits publish–subscribe patterns and message based asynchronous execution.**

Keywords—Meta-models; Behavioral semantics; Process enactment; Model Driven Engineering

I. INTRODUCTION

Modeling languages are essential artifacts in Information Systems engineering. New requirements, e.g. domain specific modeling and Model-Driven Engineering (MDE), have increased the demand for such languages in combination with adequate software tool for model manipulation, i.e. editing, transformation, verification, validation, etc. Meta-CASE and languages workbenches are leveraged to design and build such tools. Meta-models are at the core of these environments, they are used to define modeling languages abstract and concrete syntax [1]. However, meta-models do not fully describe model semantics [2] and additional techniques are necessary. For a process modeling language with concepts such as data flow, state and transition, semantics are inherently behavioral as they express the manner by which a process behave it is enacted [3]. The integration of behavioral semantics specifications into meta-models would facilitate understanding, analysis and validation of modeling languages definitions and would leverage automatic tool construction [4]. It is, however, a challenging issue as structure meta-modeling languages, e.g. MOF (Meta Object Facility), although complex endeavors by themselves, need to be complemented with other notations and formalisms. Beside, process model semantics includes two ways interaction with human actors and external software applications.

Process enactment is a fundamental issue in both business and software process modeling. Notations based on Petri nets have been used extensively either directly to model and enact business processes [5], or to express behavioral semantics for other process modeling languages, e.g. BPMN [6] and SPEM [7]. Our work differs from these approaches as its focus is goal oriented process modeling notations. We seek to express graphically and in a declarative manner the behavioral semantics for a specific goal oriented modeling notation, i.e. MAP [8], and to derive an enactment engine.

The goal of this work is to leverage MDE principles to construct software tool and develop a meta-modeling approach that bridges the gap between process semantics as conceptually perceived by the language designer, and process execution logic as implemented on computational platforms. Hence, we seek answers to the following research questions:

1. How to specify, in an explicit manner, process execution semantics at the meta-modeling level of abstraction?
2. How to exploit such specifications to build, in a full – or partial – automated manner, process enactment engine?

II. PROPOSAL

Our approach captures the execution semantics of a process modeling language directly at the meta-model level using a graphical notation (Fig. 1). We have introduced in [9] and [10] this notation which is grounded in the *Remora* event based modeling formalism [11]. Using this notation, the language designer defines a behavior meta-model besides the traditional structure meta-model. This event oriented schema expresses the language semantics in an operational manner, i.e. as state changes and algorithmic computations on the elements of the abstract syntax (i.e. structure meta-model). In fact, because of its operational nature, this behavior meta-model can be considered as the specification of a process enactment engine. As it expresses a dynamic and interactive execution logic, we target the publish/ subscribe development patterns as building block for specifying the architecture of the target engine. Publish/subscribe patterns are recognized as the paradigm of choice to develop reactive application with asynchronous interactions [12]. The final code for the enactment engine can then be obtained using an existing UML code generator.

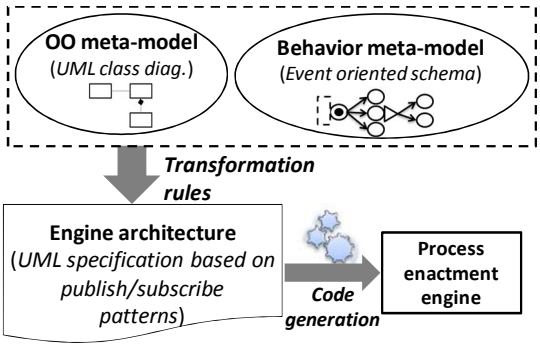

Fig. 1. Proposal overview.

Fig. 2 presents one of the transformation rules (for external interactions) that we are actually implementing and formalizing using the ATL language [13].

This approach is being applied on the case of the intentional process model MAP [8]. MAP process models capture the goals that a business or engineering process is expected to fulfill, together with a set of available strategies to realize these goals. Fig. 3 shows the behavior meta-model and the execution semantics for the MAP. At the center of this specification is the class "SectionInstance". A Map is executed in an iterative manner, one section (i.e. two intentions linked with a strategy) at a time. Each time a section is executed (i.e. section state change from *selected* to *executed*), a new set of candidate sections is computed and displayed to the user. A new enactment cycle begins when the user, i.e. MapActor, selects a candidate section for execution.

The application of this approach on the MAP confirmed the feasibility and relevance of this approach in describing language semantics and deriving a complete and detailed engine architecture with full interactive capabilities.

III. CONCLUSION

The main innovation of the meta-modeling approach presented here is the explicit representation of semantics execution in a graphical and partially declarative manner. From this specification, it is possible to derive an enactment engine using transformation rule that adequately exploit asynchronous execution patterns. Moreover, this Model Driven Engineering approach enhances significantly maintainability and portability issues when building enactment engine. Indeed, any evolution in the process modeling language will correspond to changes in the meta-models which can be propagated to the engine architecture by applying the transformation rules and the code generation steps. Compared to previous efforts in relation with MAP enactment [14] and to similar approaches in the literature [15], this method brings new insights to the realm of modeling languages design.

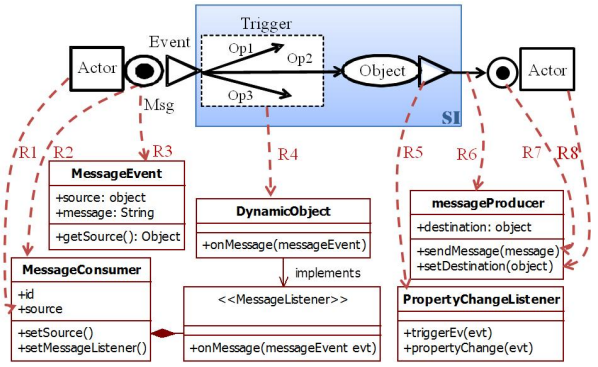

Fig. 2. An example of a transformation rule

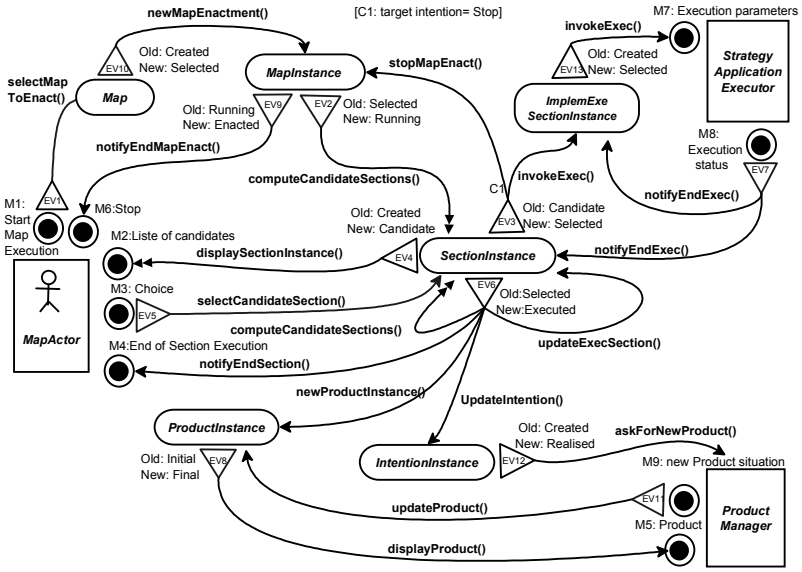

Fig. 3. Dynmaic behavior schema for the MAP modeling language

REFERENCES

[1] J.-M. Favre, D. Gasević, R. Lammel, A. Winter, "Guest Editors' Introduction to the Special Section on Software Language Engineering", IEEE Transactions on Software Eng., vol. 35, no 6, p. 737‑741, 2009.

[2] J. Sprinkle, B. Rumpe, H. Vangheluwe, G. Karsai, " Metamodelling", in *Model-Based Engineering of Embedded Real-Time Systems*, H. Giese, G. Karsai, et al. Éd., Springer, 2011, p. 57-76.

[3] B. Combemale, S. Rougemaille, X. Crégut, F. Migeon, M. Pantel, C. Maurel, B. Coulette, "Towards rigorous metamodeling", in *2nd Int. Workshop on Model-Driven Enterprise IS*, Paphos, Cyprus, 2006.

[4] B. Bryant, J. Gray, M. Mernik, P. Clarke, R. France, G. Karsai, "Challenges and Directions in Formalizing the Semantics of Modeling Languages", *Computer Science and IS*, vol.8, no 2, p. 225‑253, 2011.

[5] W. M. P. van der Aalst, A. H. M. ter Hofstede, "YAWL: yet another workflow language", *Inf. Systems*, vol. 30, n° 4, p. 245-275, 2005.

[6] R. M. Dijkman, M. Dumas, C. Ouyang, "Semantics and analysis of business process models in BPMN", *Information and Software Technology*, vol. 50, n° 12, p. 1281-1294, 2008.

[7] R. Bendraou, B. Combemale, X. Cregut, M.-P. Gervais, « Definition of an Executable SPEM 2.0 », in *Proceedings 14th Asia-Pacific Software Engineering Conf. (APSEC 2007)*, 2007, p. 390 -397.

[8] C. Rolland, N. Prakash, A. Benjamen, « A Multi-Model View of Process Modelling », *Requirements Engineering*, vol. 4, n° 1, p. 169-187, 1999.

[9] S. Mallouli, S. Assar, C. Souveyet, " Pour une perspective comportementale dans les méta-modèles de processus", *Proceedings 27th INFORSID conference*, Lille, France, 2011.

[10] S. Mallouli, S. Assar, C. Souveyet, "A Model-driven Approach to Process Enactment" *Proceedings 7th Int. Conf. on Software Paradigm Trends, ICSOFT*, 2012, pp. 351–354.

[11] C. Rolland, O. Foucault, G. Benci, *Conception des systèmes d'information: la méthode REMORA*. Paris, France: Eyrolles, 1988.

[12] P.T. Eugster, P.A. Felber, et al., "The many faces of publish/subscribe" *ACM Computing Surveys*. Vol. 35, no 2, 2003, p. 114–131.

[13] F. Jouault, I. Kurtev, "Transforming Models with ATL" *Satellite Events at MoDELS 2005 Conf. Springer*, Springer , 2006, pp. 128–138.

[14] F. Velez, "MAPExecutor: A Dynamic Enactment Support to Specify and Execute Methods with Maps", in *REFSQ'02*, Essen, Germany, 2002.

[15] R. Bendraou, J.-M. Jezéquél, F. Fleurey, "Achieving process modeling and execution through the combination of aspect and model-driven engineering approaches", J. *of Software: Evolution and Process*, vol. 24, n° 7, p. 765–781, 2012.